AI Assisted Coding

Lab Assignment 6.5

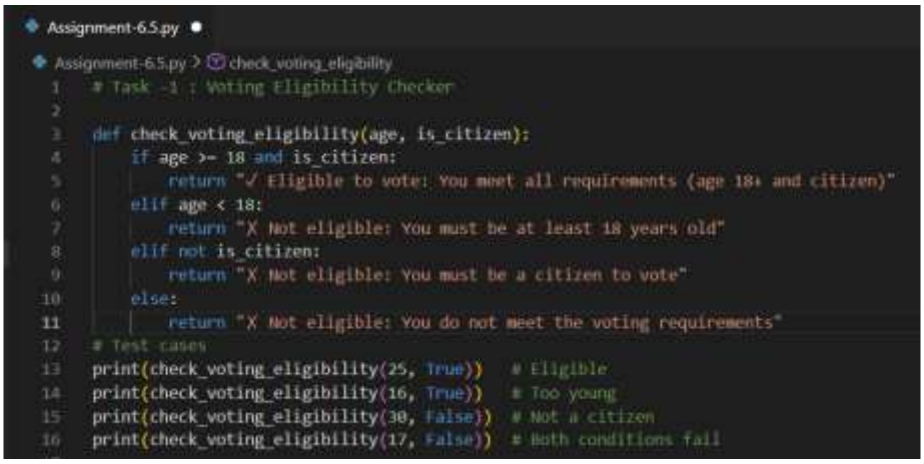
Name : K. Charan Yadav

Hall Ticket no : 2303A52367

Batch No : 20

Task -1:

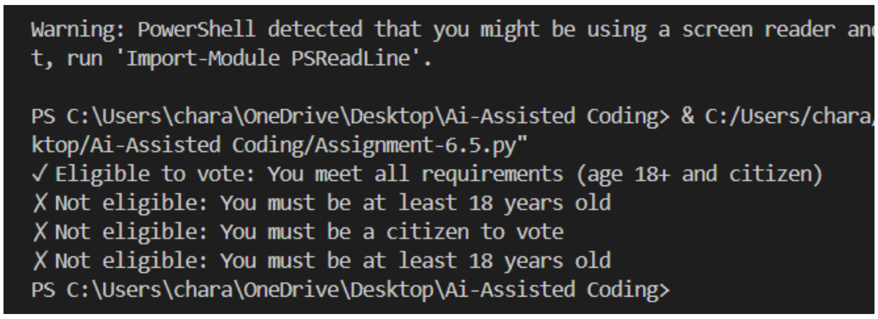
Prompt: Generate Python code to check voting eligibility based on age and citizenship



Assignment-6.5.py > check_voting_eligibility

```python
# Task -1 : Voting Eligibility Checker

def check_voting_eligibility(age, is_citizen):
    if age >= 18 and is_citizen:
        return "✓ Eligible to vote: You meet all requirements (age 18+ and citizen)"
    elif age < 18:
        return "X Not eligible: You must be at least 18 years old"
    elif not is_citizen:
        return "X Not eligible: You must be a citizen to vote"
    else:
        return "X Not eligible: You do not meet the voting requirements"
# Test cases
print(check_voting_eligibility(25, True))   # Eligible
print(check_voting_eligibility(16, True))   # Too young
print(check_voting_eligibility(30, False))  # Not a citizen
print(check_voting_eligibility(17, False))  # Both conditions fail
```

OUTPUT :



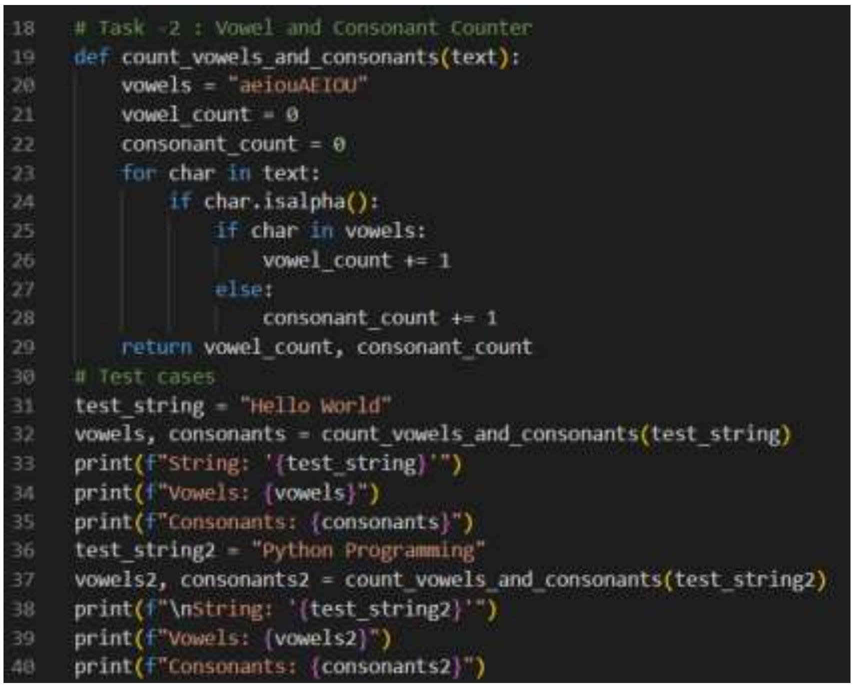
```
Warning: PowerShell detected that you might be using a screen reader an
t, run 'Import-Module PSReadLine'.

PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding> & C:/Users/chara
ktop/Ai-Assisted Coding/Assignment-6.5.py"
✓Eligible to vote: You meet all requirements (age 18+ and citizen)
✗Not eligible: You must be at least 18 years old
✗Not eligible: You must be a citizen to vote
✗Not eligible: You must be at least 18 years old
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
```

Justification:

- ✓ This program checks voting eligibility based on **age** and **citizenship** using conditional statements.
- ✓ If the person is **18 or older and a citizen**, they are eligible to vote; otherwise, the program clearly states the reason for ineligibility.
- ✓ It ensures correct decision-making with simple and readable logic.

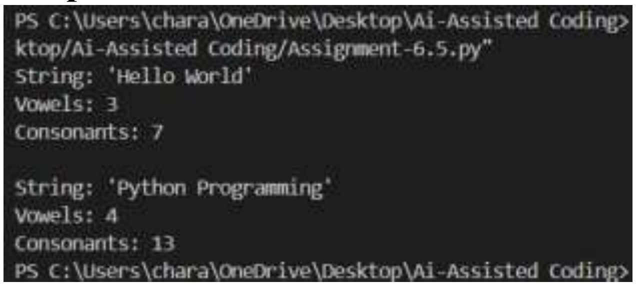
Task 2:

Prompt: Generate Python code to count vowels and consonants in a string using a loop



```python
# Task -2 : Vowel and Consonant Counter
def count_vowels_and_consonants(text):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0
    for char in text:
        if char.isalpha():
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1
    return vowel_count, consonant_count
# Test cases
test_string = "Hello World"
vowels, consonants = count_vowels_and_consonants(test_string)
print(f"String: '{test_string}'")
print(f"Vowels: {vowels}")
print(f"Consonants: {consonants}")
test_string2 = "Python Programming"
vowels2, consonants2 = count_vowels_and_consonants(test_string2)
print(f"\nString: '{test_string2}'")
print(f"Vowels: {vowels2}")
print(f"Consonants: {consonants2}")
```

Output:



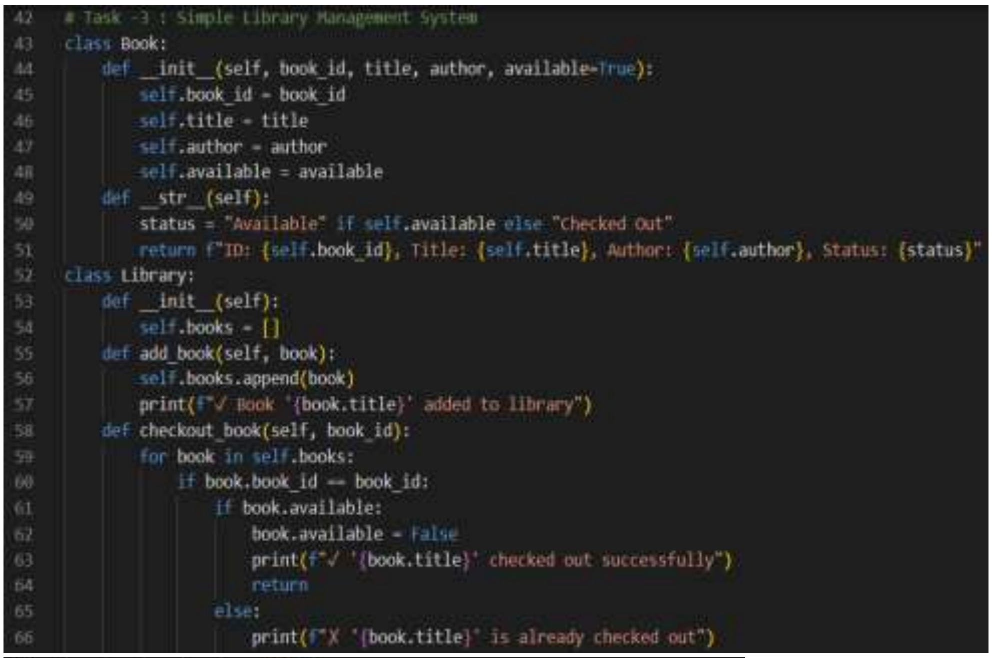
```
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
ktop/Ai-Assisted Coding/Assignment-6.5.py"
String: 'Hello World'
Vowels: 3
Consonants: 7

String: 'Python Programming'
Vowels: 4
Consonants: 13
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
```

Justification:

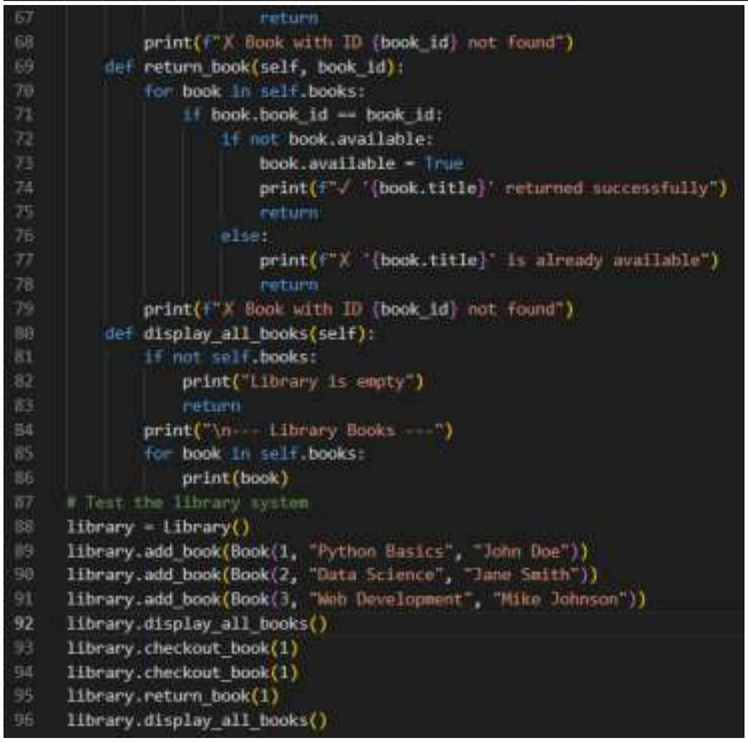
- ✓ This program counts **vowels and consonants** in a given text by iterating through each character.
- ✓ It checks only **alphabetic characters** and classifies them as vowels or consonants using conditional logic.
- ✓ The function returns accurate counts, ignoring spaces and special characters.

Task 3:

Prompt : Generate a Python program for a library management system using classes, loops, and conditional statements

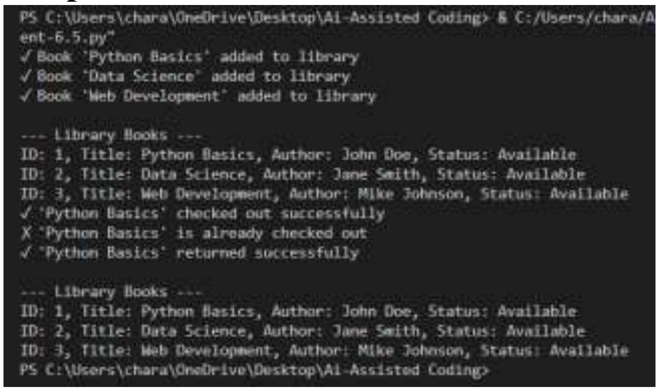


```python
# Task -3 : Simple Library Management System
class Book:
    def __init__(self, book_id, title, author, available=True):
        self.book_id = book_id
        self.title = title
        self.author = author
        self.available = available
    def __str__(self):
        status = "Available" if self.available else "Checked Out"
        return f"ID: {self.book_id}, Title: {self.title}, Author: {self.author}, Status: {status}"
class Library:
    def __init__(self):
        self.books = []
    def add_book(self, book):
        self.books.append(book)
        print(f"✓ Book '{book.title}' added to library")
    def checkout_book(self, book_id):
        for book in self.books:
            if book.book_id == book_id:
                if book.available:
                    book.available = False
                    print(f"✓ '{book.title}' checked out successfully")
                    return
                else:
                    print(f"X '{book.title}' is already checked out")
```



```python
                    return
        print(f"X Book with ID {book_id} not found")
    def return_book(self, book_id):
        for book in self.books:
            if book.book_id == book_id:
                if not book.available:
                    book.available = True
                    print(f"✓ '{book.title}' returned successfully")
                    return
                else:
                    print(f"X '{book.title}' is already available")
                    return
        print(f"X Book with ID {book_id} not found")
    def display_all_books(self):
        if not self.books:
            print("Library is empty")
            return
        print("\n--- Library Books ---")
        for book in self.books:
            print(book)
# Test the library system
library = Library()
library.add_book(Book(1, "Python Basics", "John Doe"))
library.add_book(Book(2, "Data Science", "Jane Smith"))
library.add_book(Book(3, "Web Development", "Mike Johnson"))
library.display_all_books()
library.checkout_book(1)
library.checkout_book(1)
library.return_book(1)
library.display_all_books()
```

Output :



```
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding> & C:/Users/chara/A
ent-6.5.py"
✓ Book 'Python Basics' added to library
✓ Book 'Data Science' added to library
✓ Book 'Web Development' added to library

--- Library Books ---
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
ID: 3, Title: Web Development, Author: Mike Johnson, Status: Available
✓ 'Python Basics' checked out successfully
X 'Python Basics' is already checked out
✓ 'Python Basics' returned successfully

--- Library Books ---
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
ID: 3, Title: Web Development, Author: Mike Johnson, Status: Available
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
```

Justification:

- ✓ This program implements a simple Library Management System using object-oriented programming concepts.
- ✓ The Book class stores book details and availability, while the Library class manages adding, issuing, returning, and displaying books.
- ✓ It ensures proper tracking of book status with clear messages for each operation.

Task 4 :

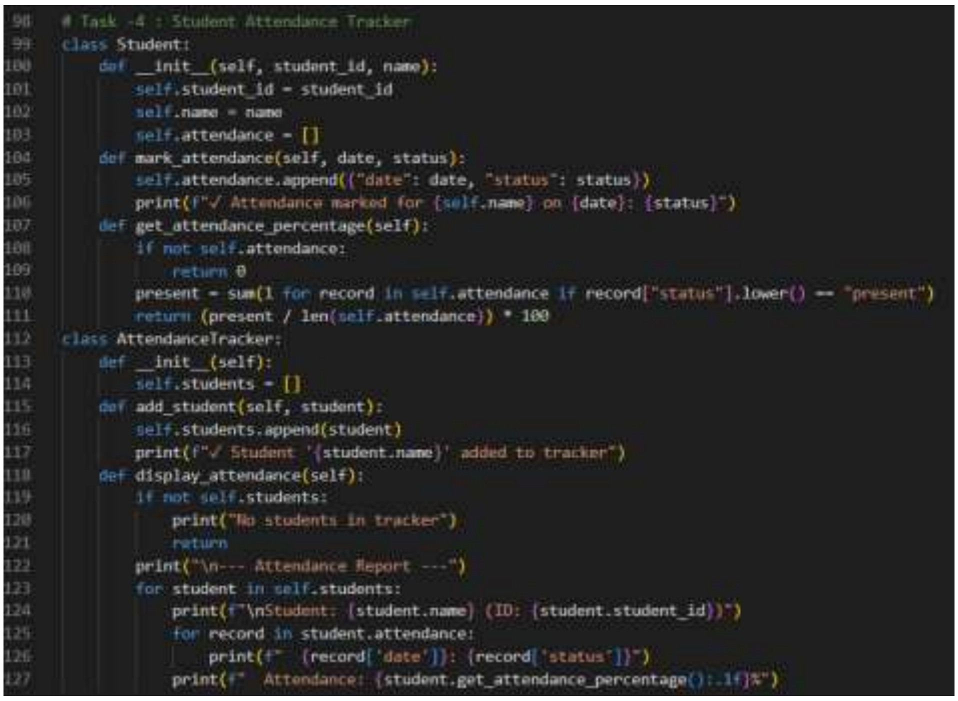
Prompt : Generate a Python class to mark and display student attendance using loops.”

Expected Output:

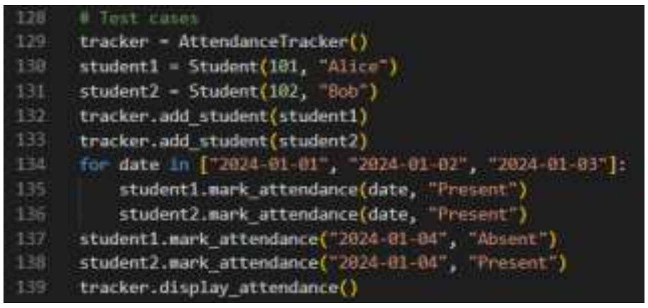
• AI-generated attendance logic.

• Correct display of attendance.

• Test cases

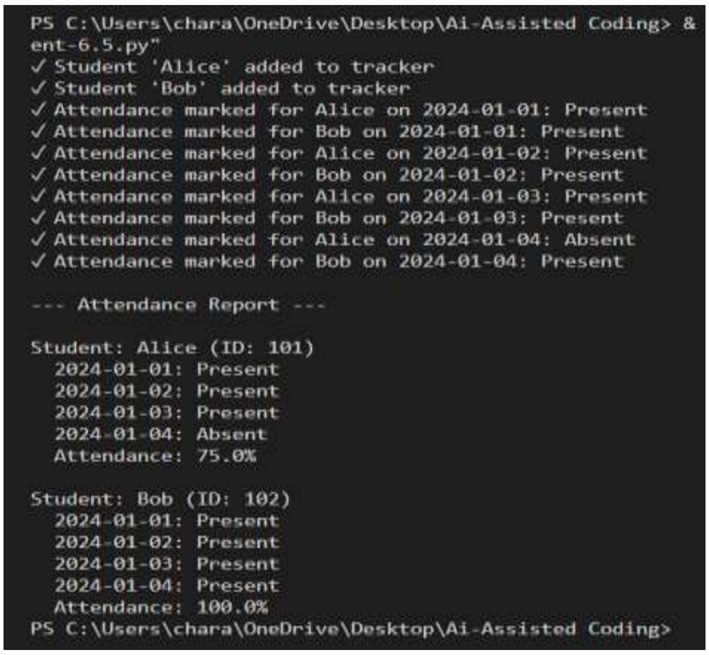


```python
# Task -4 : Student Attendance Tracker
class Student:
    def __init__(self, student_id, name):
        self.student_id = student_id
        self.name = name
        self.attendance = []
    def mark_attendance(self, date, status):
        self.attendance.append({"date": date, "status": status})
        print(f"✓ Attendance marked for {self.name} on {date}: {status}")
    def get_attendance_percentage(self):
        if not self.attendance:
            return 0
        present = sum(1 for record in self.attendance if record["status"].lower() == "present")
        return (present / len(self.attendance)) * 100
class AttendanceTracker:
    def __init__(self):
        self.students = []
    def add_student(self, student):
        self.students.append(student)
        print(f"✓ Student '{student.name}' added to tracker")
    def display_attendance(self):
        if not self.students:
            print("No students in tracker")
            return
        print("\n--- Attendance Report ---")
        for student in self.students:
            print(f"\nStudent: {student.name} (ID: {student.student_id})")
            for record in student.attendance:
                print(f"  {record['date']}: {record['status']}")
            print(f"  Attendance: {student.get_attendance_percentage():.1f}%")
```



```python
# Test cases
tracker = AttendanceTracker()
student1 = Student(101, "Alice")
student2 = Student(102, "Bob")
tracker.add_student(student1)
tracker.add_student(student2)
for date in ["2024-01-01", "2024-01-02", "2024-01-03"]:
    student1.mark_attendance(date, "Present")
    student2.mark_attendance(date, "Present")
student1.mark_attendance("2024-01-04", "Absent")
student2.mark_attendance("2024-01-04", "Present")
tracker.display_attendance()
```

Output :



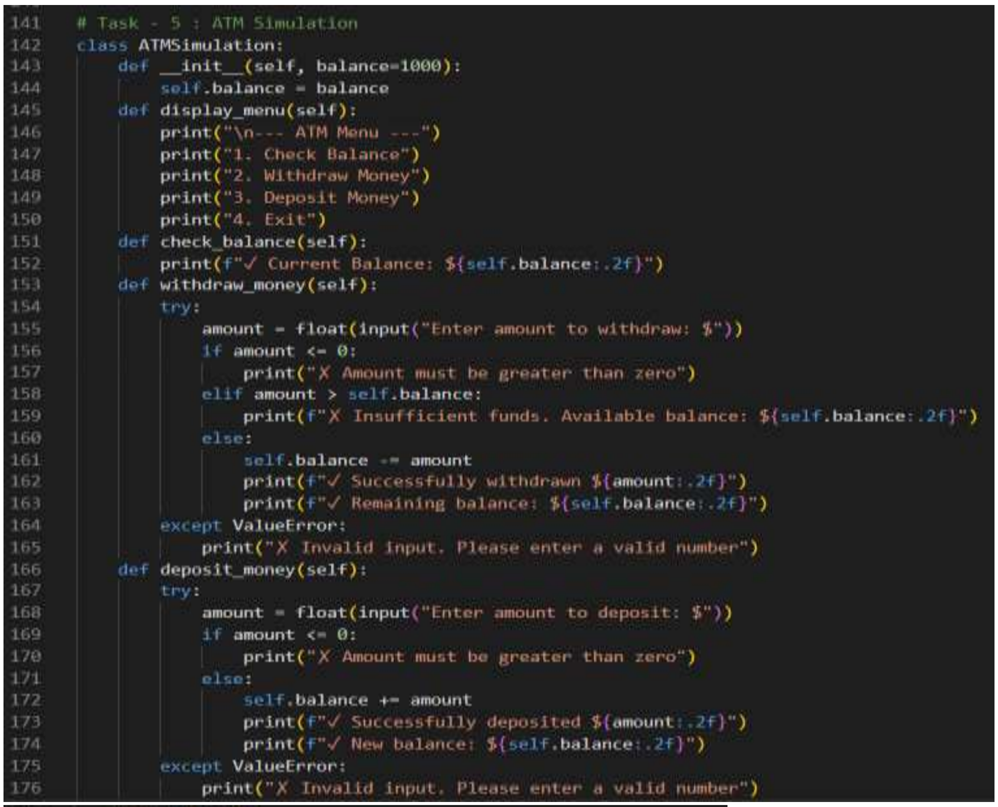
```
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding> &
ent-6.5.py"
✓ Student 'Alice' added to tracker
✓ Student 'Bob' added to tracker
✓ Attendance marked for Alice on 2024-01-01: Present
✓ Attendance marked for Bob on 2024-01-01: Present
✓ Attendance marked for Alice on 2024-01-02: Present
✓ Attendance marked for Bob on 2024-01-02: Present
✓ Attendance marked for Alice on 2024-01-03: Present
✓ Attendance marked for Bob on 2024-01-03: Present
✓ Attendance marked for Alice on 2024-01-04: Absent
✓ Attendance marked for Bob on 2024-01-04: Present

--- Attendance Report ---

Student: Alice (ID: 101)
  2024-01-01: Present
  2024-01-02: Present
  2024-01-03: Present
  2024-01-04: Absent
  Attendance: 75.0%

Student: Bob (ID: 102)
  2024-01-01: Present
  2024-01-02: Present
  2024-01-03: Present
  2024-01-04: Present
  Attendance: 100.0%
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
```

Justification:

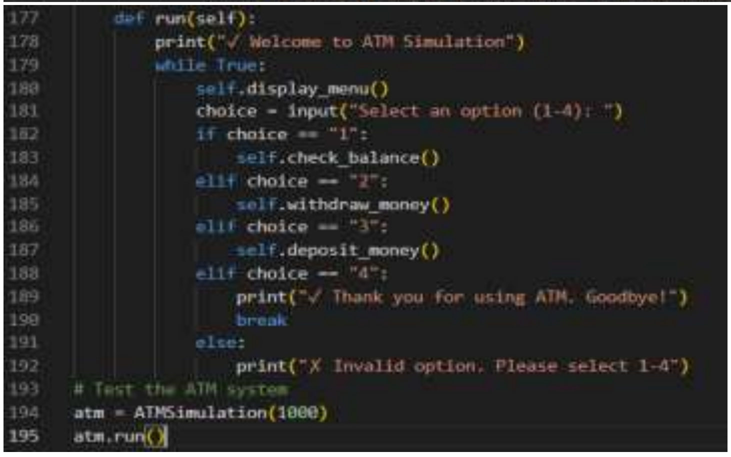
- ✓ This program tracks **student attendance** using object-oriented principles.
- ✓ The Student class records daily attendance and calculates attendance percentage, while the Attendance Tracker class manages multiple students and generates reports.
- ✓ It provides a clear and structured way to monitor attendance efficiently

<u>Task 5 :</u>

Prompt : Generate a Python program using loops and conditionals to simulate an ATM menu.

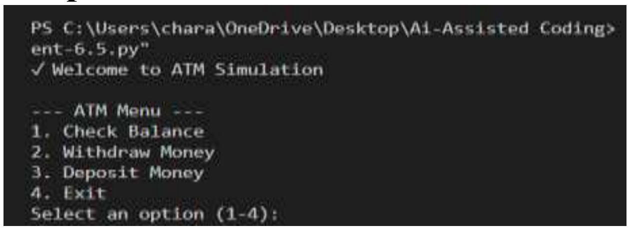


```python
# Task - 5 : ATM Simulation
class ATMSimulation:
    def __init__(self, balance=1000):
        self.balance = balance
    def display_menu(self):
        print("\n--- ATM Menu ---")
        print("1. Check Balance")
        print("2. Withdraw Money")
        print("3. Deposit Money")
        print("4. Exit")
    def check_balance(self):
        print(f"✓ Current Balance: ${self.balance:.2f}")
    def withdraw_money(self):
        try:
            amount = float(input("Enter amount to withdraw: $"))
            if amount <= 0:
                print("X Amount must be greater than zero")
            elif amount > self.balance:
                print(f"X Insufficient funds. Available balance: ${self.balance:.2f}")
            else:
                self.balance -= amount
                print(f"✓ Successfully withdrawn ${amount:.2f}")
                print(f"✓ Remaining balance: ${self.balance:.2f}")
        except ValueError:
            print("X Invalid input. Please enter a valid number")
    def deposit_money(self):
        try:
            amount = float(input("Enter amount to deposit: $"))
            if amount <= 0:
                print("X Amount must be greater than zero")
            else:
                self.balance += amount
                print(f"✓ Successfully deposited ${amount:.2f}")
                print(f"✓ New balance: ${self.balance:.2f}")
        except ValueError:
            print("X Invalid input. Please enter a valid number")
```



```python
    def run(self):
        print("✓ Welcome to ATM Simulation")
        while True:
            self.display_menu()
            choice = input("Select an option (1-4): ")
            if choice == "1":
                self.check_balance()
            elif choice == "2":
                self.withdraw_money()
            elif choice == "3":
                self.deposit_money()
            elif choice == "4":
                print("✓ Thank you for using ATM. Goodbye!")
                break
            else:
                print("X Invalid option. Please select 1-4")
# Test the ATM system
atm = ATMSimulation(1000)
atm.run()
```

Output :



```
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
ent-6.5.py"
✓ Welcome to ATM Simulation

--- ATM Menu ---
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Select an option (1-4):
```

Justification:

- ✓ This program simulates an **ATM system** that allows users to check balance, withdraw, and deposit money.
- ✓ It uses a menu-driven approach with input validation to handle invalid entries and insufficient funds.
- ✓ The system ensures secure and user-friendly banking operations through clear prompts and messages.